Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

OFFinder - Uma plataforma para divulgação de ofertas do comércio

Arthur Miranda Gomes

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:

Igor Muzetti Pereira

COORIENTAÇÃO:

Vicente José Peixoto de Amorim

Julho, 2019
João Monlevade–MG

Arthur Miranda Gomes

OFFinder - Uma plataforma para divulgação de ofertas do comércio

Orientador: Igor Muzetti Pereira

Coorientador: Vicente José Peixoto de Amorim

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Julho de 2019

G622o Gomes, Arthur Miranda.
OFFinder - Uma plataforma para divulgação de ofertas do comércio [manuscrito] / Arthur Miranda Gomes. - 2019.

51f.:

Orientador: Prof. MSc. Igor Muzetti Pereira.
Coorientador: Prof. MSc. Vicente José Peixoto de Amorim.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Android (Recurso eletrônico). 2. Plataforma Web. 3. Marketing na internet. I. Pereira, Igor Muzetti. II. Amorim, Vicente José Peixoto de. III. Universidade Federal de Ouro Preto. IV. Titulo.

CDU: 004.777

Catalogação: ficha.sisbin@ufop.edu.br

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Arthur Miranda Gomes

OFFinder - Uma plataforma para divulgação de ofertas do comércio

Monografia apresentada ao Curso de Engenharia da Computação da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 - Trabalho de Conclusão de Curso II

Aprovada em 12 de Julho de 2019.

Membros da banca

MSc. - Igor Muzetti Pereira - Orientador - Universidade Federal de Ouro Preto
Dr. Vicente José Peixoto de Amorim - Coorientador - Universidade Federal de Ouro Preto
Dr. Filipe Ribeiro Nunes - Universidade Federal de Ouro Preto
Dra. Lucinéia Souza Maia - Universidade Federal de Ouro Preto

Igor Muzetti Pereira, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 11/05/2021

Documento assinado eletronicamente por **Igor Muzetti Pereira**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 11/05/2021, às 10:22, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0169618** e o código CRC **5BEAFE41**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Este trabalho é dedicado aos meus pais, que com muito carinho, dedicação e apoio, não mediram esforços para que eu chegasse até essa etapa da minha vida.

Agradecimentos

Aos meus pais e minha madrinha, que sempre me apoiaram e foram exemplos de determinação.

Agradeço aos meus amigos, que fizeram parte dessa longa jornada, e que espero que continuem presentes em minha vida.

Ao meu orientador, por ter acreditado em minha capacidade de realização do trabalho, e por toda sua paciência e apoio.

E a todos aqueles, que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

"You, me, or nobody is gonna hit as hard as life. But it ain't about how hard you hit. It's about how hard you can get hit and keep moving forward; how much you can take and keep moving forward. That's how winning is done!"

— Sylvester Stallone, Rocky Balboa,
in: Rocky Balboa.

Resumo

Este trabalho teve como proposito desenvolver uma plataforma online para divulgação de ofertas, capaz de atender as necessidades de pequenos, médios e grandes comerciantes. As ofertas são divulgadas como encartes virtuais e construídas na própria plataforma. Para os clientes, a plataforma deve ser vista como um ambiente intuitivo, com recursos que facilitem encontrar suas ofertas de interesse. O processo de construção desta plataforma é baseado em técnicas de desenvolvimento ágil, e visa utilizar tecnologias presentes no mercado atual.

Palavras-chave: Android. Platforma Web. Marketing digital.

Abstract

The purpose of this work was to develop an online platform for the dissemination of offers, capable of meeting the needs of small, medium and large merchants. The offers are disclosed as virtual pamphlets and built on the platform itself. For customers, the platform should be seen to be an intuitive environment, with features that make it easy to find your offers of interest. The process of building this platform is based on agile development techniques, and aims to use technologies present in the current market.

Key-words: Android. Web Platform. Digital marketing.

Lista de ilustrações

Figura 1 – Aplicativo Guiato 19
Figura 2 – Plataforma AondeConvem 20
Figura 3 – Plataforma Tiendeo 20
Figura 4 – Estrutura da história de usuário 22
Figura 5 – Especificação da historia "Autenticar usuário" 22
Figura 6 – Padrão para definição para os critérios de aceitação 22
Figura 7 – Exemplo de teste para o cenário de sucesso da funcionalidade "Autenticar usuário" 23
Figura 8 – Design inicial das telas plataforma 23
Figura 9 – Visão Geral da Arquitetura da Aplicação 27
Figura 10 – Arquitetura Geral da Aplicação e Suas Comunicações 29
Figura 11 – Documentação da API para controladora entidade oferta 31
Figura 12 – Documentação da API para controladora entidade loja 32
Figura 13 – Documentação da API para controladora da relação loja-oferta 33
Figura 14 – Documentação da API para controladora da relação loja-produto . . . 33
Figura 15 – Documentação da API para controladora da relação loja-promoção . . 34
Figura 16 – Documentação da API para controladora da relação oferta-promoção . 35
Figura 17 – Documentação da API para controladora da relação produto-categoria 36
Figura 18 – Documentação da API para controladora da relação promoção-produto 37
Figura 19 – Sucesso - Acessar a aplicação com credenciais (e-mail e senha) validos, e previamente cadastrados. 40
Figura 20 – Erro - Acessar a aplicação com e-mail inválido. 41
Figura 21 – Erro - Acessar a aplicação com senha inválida. 42
Figura 22 – Erro - Acessar a aplicação por usuário não cadastrado. 43
Figura 23 – Tela de registro de novo usuário 44
Figura 24 – Tela de cadastro de ofertas e de produtos 45
Figura 25 – Tela de listagem de produtos cadastrados 46
Figura 26 – Tela principal da plataforma web 47
Figura 27 – Tela de exibição de ofertas válidas 47

Lista de abreviaturas e siglas

API	*Application Programming Interface* (Interface de programação de aplicativos)
SPA	*Single Page Application* (Aplicativo de página única)
BDD	*Behaviour Driven Development* (Desenvolvimento Orientado por Comportamento)
SDK	*Software development kit* (Kit de desenvolvimento de software)
REST	*Representational State Transfer* (Transferência Representacional de Estado)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)

Sumário

1 INTRODUÇÃO . . . 15
1.1 O problema de pesquisa . . . 15
1.2 Objetivos . . . 15
1.3 Metodologia . . . 16
1.4 Organização do trabalho . . . 16

2 REVISÃO BIBLIOGRÁFICA . . . 17
2.1 Engenharia de Software . . . 17
2.1.1 Desenvolvimento Ágil . . . 17
2.1.2 *Behaviour Driven Development* . . . 18
2.1.3 *Back-End* . . . 18
2.1.3.1 API REST . . . 18
2.1.4 Front-End . . . 19
2.2 Trabalhos Correlatos . . . 19

3 DESENVOLVIMENTO . . . 21
3.1 Métodos . . . 21
3.2 Ferramentas . . . 23
3.2.1 Loopback . . . 24
3.2.2 React . . . 24
3.2.3 Flutter . . . 25
3.2.3.1 Bloc . . . 25
3.2.4 Docker . . . 25
3.2.4.1 Imagens e Contêineres . . . 26
3.2.4.2 Contêineres e Máquinas Virtuais . . . 26
3.2.5 Firebase . . . 26
3.2.5.1 Firebase Auth . . . 26
3.2.5.2 Firebase Storage . . . 27
3.3 Arquitetura . . . 27
3.4 Codificação . . . 29
3.4.1 API . . . 29
3.4.1.1 *Endpoints* . . . 30
3.4.1.1.1 OfertasController . . . 30
3.4.1.1.2 LojasController . . . 31
3.4.1.1.3 LojaOfertaController . . . 32
3.4.1.1.4 LojaProdutoController . . . 33

3.4.1.1.5 LojaPromocaoController 34
3.4.1.1.6 OfertaPromocaoController 35
3.4.1.1.7 ProdutoCategoriaController 36
3.4.1.1.8 PromocaoProdutoController 37
3.4.2 Cliente Web 38
3.4.3 Aplicação para Administradores 38

4 RESULTADOS 39

5 CONCLUSÃO 48
5.1 Considerações Finais 48
5.2 Trabalhos Futuros 48

REFERÊNCIAS 50

1 Introdução

1.1 O problema de pesquisa

O problema surge na dificuldade de pequenos comerciantes realizarem a divulgação de suas promoções, considerando o custo da confecção de materiais gráficos, como panfletos e catálogos, ou gastos com divulgação em jornais, anúncios ou *outdoors* e a necessidade de um alvará para sua distribuição. Além do fator financeiro, comerciantes que apenas utilizam o *marketing* tradicional estão limitados a apenas uma área atingida, como bairros ou cidades (MORAIS, 2013). O *marketing* digital surge como um alternativa e complemento ao marketing tradicional, sendo definido como um conjunto de informações e ações que podem ser feitas em diversos meios digitais com o objetivo de promover empresas e produtos (RESULTADOS DIGITAIS, 2016). O *marketing* digital é altamente eficiente em custos (ABRANTES, 2016) porém ainda exige certo conhecimento técnico para sua implantação. Assim, uma plataforma que facilite a divulgação de propagandas é algo extremamente relevante para pequenos comerciantes, tornando significativo o escopo deste projeto.

Existem diversas aplicações que disponibilizam encartes de ofertas do comércio na WEB, como grandes exemplos de empresas desse ramo temos: Guiato, AondeConvem e Tiendeo. Estas empresas citadas possuem como característica em comum a construção de suas ofertas a partir de material publicitário físico, o qual é escaneado para a plataforma e disponibilizado para seus clientes através de sites e/ou aplicativos. Esta metodologia para construção de ofertas possui como principal vantagem a simplicidade do processo e o reaproveitamento de material existente. Porém esta metodologia é apenas uma ferramenta auxiliar no processo de divulgação de panfletos, uma vez que é necessário a existência do material gráfico para a exibição no site, o que não incentiva a redução da produção de material gráfico físico.

1.2 Objetivos

Este trabalho tem como objetivo a criação de uma plataforma de divulgação de promoções capaz de fornecer um ambiente de interação entre comerciantes e clientes, permitindo a divulgação das principais ofertas de forma simples e rápida. Para os clientes, esta plataforma deve ser vista como um ambiente intuitivo para encontrarem seus produtos de interesse. O cadastro das promoções é realizado na plataforma pelo próprio comerciante. Os produtos em promoção de um determinado comércio devem ser exibidos para os clientes que acessarem ao site de acordo com sua pesquisa.

Este trabalho possui os seguintes objetivos específicos:

- Pesquisar as principais tecnologias do mercado de desenvolvimento.
- Trabalhar sobre padrões de *design* de aplicações com foco em escalabilidade e manutenibilidade.
- Adotar técnicas de desenvolvimento ágil no processo de desenvolvimento.
- Construir uma aplicação completa arquiteturalmente.

1.3 Metodologia

O objeto de pesquisa deste trabalho é o processo de construção de *software* baseado em técnicas de desenvolvimento ágil e o processo de elaboração da arquitetura de uma aplicação, tendo em vista as principais tecnologias disponíveis no mercado durante sua construção.

Os passos para execução deste trabalho são assim definidos:

- Pesquisa por Soluções Correlatas
- Elaboração do *Product Backlog*
- Elaboração das *User Stories*
- Elaboração dos Cenários de Teste
- Elaboração da Arquitetura da Solução
- Codificação
- Testes baseados nos Cenários Previstos
- Validação

1.4 Organização do trabalho

Este trabalho está dividido em cinco capítulos. O Capítulo 1 introduz o conceito de *Marketing Digital* e justifica a importância deste trabalho. O Capítulo 2 apresenta trabalhos correlatos que serviram de fundamentação teórica. No Capítulo 3 são apresentados os requisitos e métodos utilizados para o desenvolvimento da plataforma de divulgação de ofertas. O Capítulo 4 discute metodologias de avaliação e resultados obtidos e, por fim, são apresentadas as conclusões no Capítulo 5.

2 Revisão bibliográfica

Este capítulo apresenta a fundamentação teórica do projeto, além de mostrar alguns produtos já existentes no mercado, que possuem características semelhantes ao projeto desenvolvido.

2.1 Engenharia de Software

O processo de criação de um *software* abrange uma variedade de técnicas que visam a organização, produtividade e qualidade durante o seu desenvolvimento e manutenção. Nesse sentido a engenharia de *software* é adotada a fim de garantir essas qualidades ao processo e ao produto.

A engenharia de requisitos, é uma área da engenharia de *software*, que possui papel fundamental na criação do *software*, sendo responsável por alinhar as ideias iniciais do projeto com o seu desenvolvimento, fornecendo uma espécie de guia para o desenvolvedor.

2.1.1 Desenvolvimento Ágil

Nos últimos anos as abordagens ágeis de desenvolvimento de *software* se tornaram mais populares. Assim, vários métodos foram desenvolvidos com o objetivo de acelerar o processo de desenvolvimento e garantir que o software atendas às necessidades de mudança propostas pelo cliente (PAETSCH; EBERLEIN; MAURER, 2003).

Dentre as metodologias ágeis, as mais populares são o XP, o Scrum e o Kanban (BOWES, 2015). Essas metodologias, de um modo geral, compartilham alguns princípios comuns, focados na satisfação do cliente, adoção de requisitos em constante mudança, fornecimento assíduo de *software* funcional e colaboração frequente de pessoas de negócio e desenvolvedores (PAETSCH; EBERLEIN; MAURER, 2003).

Os princípios básicos do desenvolvimento ágil, descritos pelo *Manifesto for Agile Software Development* (BECK et al., 2001), são:

- Indivíduos e iterações mais que processos e ferramentas;
- Software funcional mais que documentação abrangente;
- Colaboração do cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

2.1.2 *Behaviour Driven Development*

A abordagem do desenvolvimento dirigido por comportamento (*Behaviour Driven Development* - BDD) funciona muito bem como uma metodologia ágil que visa priorizar o comportamento do *software*. Essa abordagem estimula os membros do projeto a escreverem o sistema se baseando em seus comportamentos, que por sua vez devem estar definidos como forma de requisitos.

Desta forma, ao adotarmos o BDD, nossa primeira preocupação é definir os requisitos do sistema. Um conjunto de requisitos é organizado em grupos que juntos, formam um serviço que o software desempenhará (PEREIRA, 2014). Este conjunto de serviços forma o *product backlog*, que é um dos registros base da documentação do processo de construção do software e que define os principais requisitos daquele sistema. Além disso, a partir do *product backlog* é construído outros documentos que auxiliam na compreensão dos requisitos do sistema, como por exemplo o *selected backlog* e a definição dos cenários de teste.

2.1.3 *Back-End*

Na estrutura de um sistema computacional, o *back-end* é a camada que não interage diretamente com os usuários. Esta camada abrange os servidores e as regras de negócio envolvidas no fluxo de execução do sistema. O *back-end* muitas vezes deve ser capaz de interagir com o *front-end* ou com outras aplicações, e para isso é comum a criação de *API's*.

2.1.3.1 API REST

Uma API (*Application Programming Interface*), consiste em um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na WEB e possui um papel importante no ciclo de vida de um sistema. A partir de uma API é possível expor um conjunto de dados e funções para facilitar interações entre softwares e possibilitar a troca de informações. Assim uma API é uma ferramenta que permite a interoperabilidade entre aplicações.

Um dos modelos de arquiteturas mais utilizados no projeto de arquitetura de software distribuído é o REST (*Representational State Transfer*). Este modelo foi projetado por Roy Fielding, um dos principais criadores do protocolo HTTP, e consiste em um conjunto de princípios ou regras que, quando seguidas, permite a criação de um projeto com interfaces bem definidas.

2.1.4 Front-End

Ao acessarmos uma aplicação a primeira camada a qual visualizamos é conhecida como *front-end*. Esta camada abrange a interface de interação com o usuário.

Nas ultimas décadas inúmeros *frameworks*, bibliotecas e ferramentas surgiram para auxiliar na criação de interfaces de usuário.

Com o foco na qualidade da utilização do usuário, as SPA (*Single Page Application*) tem se popularizado cada vez mais, sendo utilizadas em grandes sites, como Twitter, Gmail e Trello (GIL, 2016). Esse tipo de aplicação possui como principal característica exibir em uma única página todo conteúdo de uma aplicação.

A construção de uma SPA's muitas vezes é realizada com auxílio de bibliotecas e *frameworks*, como por exemplo React, Angular e Vue.js.

2.2 Trabalhos Correlatos

Atualmente existem algumas aplicações destinadas a divulgação de forma virtual de panfletos, folhetos e encartes. Para fins de exemplo, algumas delas são apresentadas:

A Guiato (GUIATO, 2018) é parte do Grupo Bonial.com, a principal rede internacional de publicidade móvel baseada em localização. Desde o seu lançamento na Alemanha em 2008, o grupo cresceu chegando a 13 mercados em todo o mundo. Todos os meses 22 milhões de compradores se voltam para os aplicativos e sites móveis da Bonial. A Figura 1 ilustra o aplicativo Guiato.

Figura 1 – Aplicativo Guiato

Fonte: (GUIATO, 2018)

AondeConvem (AONDECONVEM, 2018) é uma plataforma de publicidade, pertencente ao grupo internacional ShopFully, utilizada por mais de 30 milhões de consumidores em todo o mundo. A Figura 2 ilustra a plataforma AondeConvem.

Figura 2 – Plataforma AondeConvem

Fonte: (AONDECONVEM, 2018)

O Tiendeo (TIENDEO, 2018) é uma plataforma que agrega catálogos e folhetos de ofertas semanais de diversas lojas. Esta *startup* possui atualmente uma média de 35 milhões de visitas por mês e mais de 7 milhões de downloads do aplicativo. A Figura 3 ilustra a plataforma Tiendeo.

Figura 3 – Plataforma Tiendeo

Fonte: (TIENDEO, 2018)

3 Desenvolvimento

Este capítulo apresenta as decisões tomadas no desenvolvimento da plataforma de divulgação de ofertas e explica o seu funcionamento em detalhes. Também apresenta como a aplicação foi projetada e testada.

3.1 Métodos

Este projeto foi construído com base em técnicas de desenvolvimento ágil, seguindo os princípios do *Scrum*. E cada uma de suas etapas foram pensadas visando a capacidade de resposta, de forma ágil, às possíveis mudanças de requisitos que poderiam ocorrer durante todo o processo.

O ponto inicial do projeto foi o levantamento dos requisitos da plataforma. Para tanto foi utilizada a abordagem de desenvolvimento dirigido por comportamento BDD. Assim a primeira etapa do projeto consistiu na construção de um *Product Backlog* o qual foi armazenado na pagina da Dokuwiki do projeto. O *product backlog* então, contém um conjunto de itens que são agrupados em serviços ou componentes e cada serviço ou componente é formado por um conjunto de itens do *product backlog*, que são os requisitos do *software*. (PEREIRA, 2014).

Para este projeto o *backlog* inicial continha os seguintes serviços básicos:

- Serviço de Gerenciamento do Usuário
- Serviço de Gerenciamento de Ofertas
- Serviço de Gerenciamento de Produtos
- Serviço de Visualização de Ofertas

Cada um dos serviços acima foi desmembrado em funcionalidades menores. Como exemplo, para o serviço de Gerenciamento do Usuário, as funcionalidades listadas foram:

- Autenticar usuário
- Registrar novo usuário
- Recuperar senha
- Editar dados do usuário

Uma vez definidos o conjunto inicial de serviços e funcionalidades básicas, entramos na etapa de escrita das histórias de usuário. Assim, seguindo o BDD, cada requisito foi descrito como uma história de usuário, a qual descreve como o software é utilizado. Para isso foi adotado a estrutura básica de história representada pela Figura 4:

Figura 4 – Estrutura da história de usuário

```
COMO [papel]
EU GOSTARIA [funcionalidade]
PARA CONSEGUIR [objetivo]
```

Fonte: o autor

Como exemplo, para a funcionalidade **Autenticar**, do serviço de **Gerenciamento de Usuário**, a história de usuário gerada é ilustrada pela Figura 5.

Figura 5 – Especificação da historia "Autenticar usuário"

```
COMO um usuário cadastrado.
EU GOSTARIA de logar no sistema mediante ao fornecimento de "email" e "senha".
PARA CONSEGUIR acesso às funcionalidades do sistema.
```

Fonte: o autor

Em seguida cada história foi decomposta em cenários de teste de forma que, obrigatoriamente, para cada uma das histórias exista pelo menos um cenário principal. Em complemento foram criados alguns cenários alternativos. Estes cenários de teste foram utilizados como base para a modelagem da arquitetura da aplicação e como sequência de etapas a serem concluídas para a construção da mesma. As Figuras 6 e ?? representam, respectivamente, a estrutura padrão de um teste como critério de aceitação, e o teste de sucesso para a funcionalidade "Autenticar usuário".

Figura 6 – Padrão para definição para os critérios de aceitação

```
Given (Dado)   : pré-condição
When (Quando)  : evento
Then (Então)   : resultado esperado
And (E)        : extensão
But (Mas)      : extensão
```

Fonte: o autor

Com o *Product Backlog* inicial escrito, e os testes de aceitação elaborados, temos então material suficiente para o inicio da elaboração do projeto.

```
DADO que um usuário cadastrado forneça "email" e "senha" válidas
QUANDO o botão "entrar" for pressionado
ENTÃO a aplicação deve transitar para a tela principal a partir da qual
o usuário poderá selecionar seus serviços
```

Figura 7 – Exemplo de teste para o cenário de sucesso da funcionalidade "Autenticar usuário"

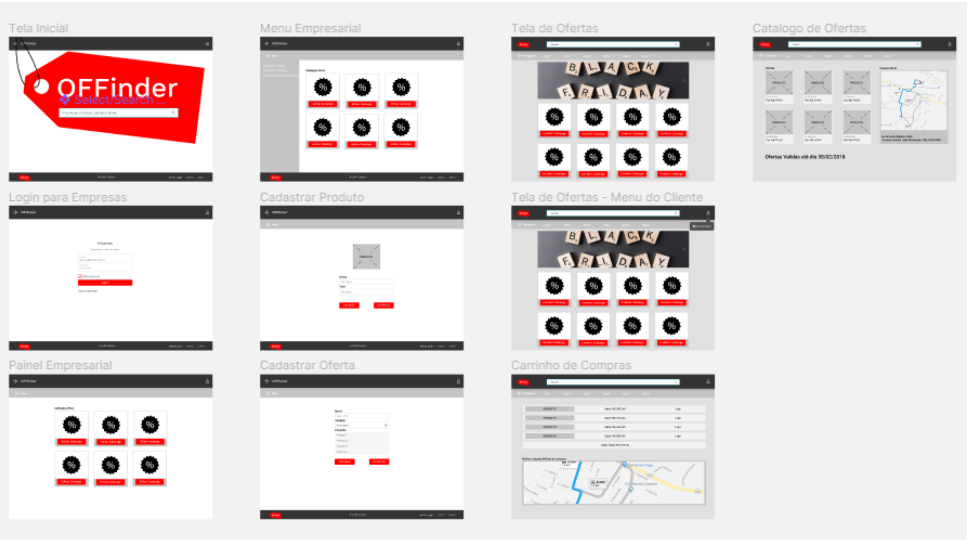
Fonte: o autor

Como este projeto foi construído com base em metodologias ágeis, em especifico o ***Scrum***, a cada uma das *sprints* (intervalo de tempo de quatro a cinco semanas) parte dos itens do *backlog* foi priorizada para ser desenvolvida.

Entretanto, a primeira *sprint* realizada após a definição do *backlog* foi focada no *design* da aplicação.

O desenho inicial do projeto foi construído na plataforma Figma, uma ferramenta de design de interface colaborativa, que permite projetar e prototipar interfaces de aplicações, de forma simples e rápida. (FIGMA, 2018). O resultado do projeto de design pode ser visto na Figura 8.

Figura 8 – Design inicial das telas plataforma



Fonte: o autor

3.2 Ferramentas

Nesta seção serão apresentadas as ferramentas básicas que compõem a arquitetura deste projeto.

3.2.1 Loopback

LoopBack é um *framework* Node.js de código aberto altamente extensível, que permite a criação de *API's REST* de uma forma dinâmica e intuitiva, sem a necessidade de uma codificação direta.

Nas construção de um back-end com *Loopback* um dos principais conceitos trabalhados é o Modelo. Um Modelo é uma representação de uma fonte de dados *back-end*, que pode ser, por exemplo, um banco de dados ou até mesmo outros serviços de *back-end*.

Um dos principais recursos fornecidos pelo Loopback, é a geração automática de uma API REST a partir do Modelo definido pelo usuário, tal API contem os principais métodos de interesse para a manipulação e consulta de dados do Modelo, como por exemplo operações de consulta, alteração, criação e exclusão.

Uma aplicação Loopback é composta por camadas, onde cada uma camada possui um conjunto de responsabilidades para compor a aplicação. Assim, a camada de dados tem papel de prover dados os quais são representados por objetos da camada de modelos. As regras da aplicação sobre os modelos estão contidas na camada de repositórios e são fornecidas por *end-points* da camada de controladoras.

O Loopback facilita o desenvolvimento de aplicações, principalmente por possuir uma ferramenta de linha de comandos poderosa, com a qual conseguimos gerar membros para cada uma das camadas que compõem a arquitetura da API. Além de Modelos, uma aplicação Loopback permite a construção de relações entre eles, de forma semelhante ao conceito de relacionamento em Banco de Dados. Desta forma é possível estabelecer, entre dois modelos, as relações um-pra-um, um-pra-muitos e muitos-pra-muitos. Dado um modelo e suas relações é possível criar consultas aninhadas na *API REST*, de forma a permitir operações em um Modelo, a partir de Modelos com o qual este possua um relacionamento.

3.2.2 React

O *React* é uma biblioteca de JavaScript para a construção de interfaces de usuário. O React surgiu em 2011, no Facebook sendo utilizada na interface do mural de notícias do site. (BRASIL, 2018)

Com o React é possível construir interfaces de usuário interativas, a partir da construção de componentes encapsulados. Um componente é um elemento de uma interface, e pode conter um estado. Quando ocorre uma manipulação de estados os componentes são renderizados novamente, e é isso que define a característica reativa do React.

3.2.3 Flutter

O Flutter é um *SDK*(Kit de desenvolvimento de software), desenvolvido pela Google para desenvolvimento hibrido de aplicações, utilizando a linguagem Dart como base de criação dos aplicativos.

3.2.3.1 Bloc

Atualmente, diversos padrões arquiteturais para a construção de aplicações Flutter tem surgido, dentre eles o padrão Block se destaca, uma vez que é recomendado pela equipe técnica da Google.

O Bloc facilita a separação da camada de apresentação da camada de lógica de negócios, aumentando assim o desempenho do código, a facilidade de testar e sua capacidade de reutilização. (ANGELOV, 2019)

Uma aplicação seguindo os princípios do padrão Bloc, deve conter as seguintes camadas básicas:

- **Camada de Dados:** A responsabilidade da camada de dados é recuperar / manipular dados de uma ou mais fontes.
  - **Provedor de Dados:** A responsabilidade do provedor de dados é fornecer dados brutos. O provedor de dados deve ser genérico e versátil.
  - **Repositório:** A camada de repositório é um *wrapper* em torno de um ou mais provedores de dados com os quais a camada Bloc se comunica.
- **Camada de Negócios (Bloc):** A responsabilidade da camada Bloc é responder aos eventos da camada de apresentação com novos estados. A camada Bloc pode depender de um ou mais repositórios para recuperar dados necessários para construir o estado do aplicativo.
- **Camada de Apresentação:** A responsabilidade da camada de apresentação é descobrir como renderizar o aplicativo com base em um ou mais estados do bloco. Além disso, ele deve manipular eventos de entrada do usuário e de ciclo de vida do aplicativo.

3.2.4 Docker

O Docker, de acordo com (DOCKER, 2018), é uma plataforma para desenvolvedores e administradores de sistemas que possui como finalidade auxiliar nas tarefas de desenvolvimento, implantação, e execução de aplicações em contêineres. A conteinerização de aplicações tem se tornado cada vez mais popular, devido as características dos contêineres:

- Flexibilidade: Até as mais complexas aplicações podem ser disponibilizadas em contêineres.
- Leveza: Os contêineres aproveitam e compartilham o *Kernel* do *Host*.
- Intercambiáveis: É possível implantar atualizações e melhorias em produção.
- Portabilidade: É possível construir sua aplicação localmente, implantar na nuvem e executar em qualquer lugar.
- Escalabilidade: É possível aumentar e automaticamente distribuir replicas de contêineres.
- Empilháveis: É possível empilhar serviços verticalmente e em produção.

3.2.4.1 Imagens e Contêineres

Um contêiner é uma instância em execução de uma imagem que por sua vez é um pacote executável que inclui tudo que é necessário para executar a aplicação, como por exemplo o código, o ambiente de execução, as bibliotecas, as variáveis de ambiente e os arquivos de configuração.

3.2.4.2 Contêineres e Máquinas Virtuais

Por padrão, um contêiner é executado nativamente sobre Linux e compartilha o *Kernel* da maquina *host* com outros contêineres. Assim ele é executado como um processo discreto que não consome mais memória que outros executáveis convencionais. Em comparação, uma máquina virtual executa um sistema operacional completo, que realiza acesso virtual aos recursos do *host*. Por possuir esta construção, em geral uma máquina virtual provê mais recursos do que o necessário para a execução de uma aplicação.

3.2.5 Firebase

O Firebase é uma plataforma de desenvolvimento de aplicativos para dispositivos móveis e da WEB. Dentre seus produtos, dois relevantes para a construção deste trabalho são o Firebase Auth e o Firebase Storage.

3.2.5.1 Firebase Auth

O Firebase Auth é um produto que permite gerenciar usuários de uma aplicação de maneira simples e segura. Este produto oferece vários métodos para autenticação, incluindo e-mail e senha, provedores de terceiros, como o Google ou o Facebook, e o uso do sistema de contas existente diretamente.

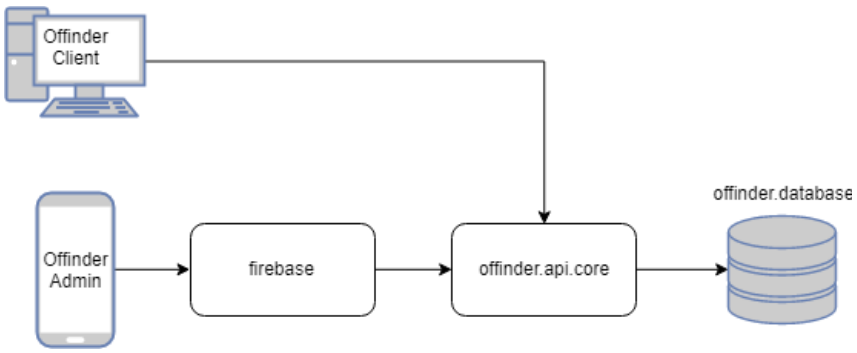
3.2.5.2 Firebase Storage

O Firebase Storage é um produto que permite o armazenamento e compartilhamento de conteúdo gerado pelo usuário, como imagens, áudio e vídeo, com armazenamento de objetos poderoso, simples e econômico construído para a escalalabilidade do Google. Os SDKs do Firebase para Cloud Storage adicionam segurança do Google a uploads e downloads de arquivos para seus aplicativos do Firebase, independentemente da qualidade da rede.

3.3 Arquitetura

O projeto inicial da arquitetura da aplicação consistia basicamente de um servidor e uma aplicação WEB. O papel do servidor seria fornecer acesso e permitir a manipulação de dados para a aplicação cliente. E a aplicação seria responsável por permitir que um administrador de uma loja, pudesse gerenciar dados referentes a suas ofertas na plataforma, e que um cliente pudesse visualizar todas as ofertas não expiradas cadastradas no site. Porém, em sua versão final, pensando na usabilidade do administrador de ofertas, o ambiente administrativo foi desvinculado do ambiente de clientes e reestruturado sobre uma aplicação móvel, a fim de permitir facilidades como por exemplo, tirar uma foto do produto a ser cadastrado e realizar *upload* da mesma no Firebase. A Figura 9 mostra as principais conexões entre a aplicação cliente web, a aplicação administrativa mobile, e os servidores de *back-end*.

Figura 9 – Visão Geral da Arquitetura da Aplicação



Fonte: o autor

O banco de dados escolhido foi o *PostgreSQL* em sua versão 11.4. Para facilitar as tarefas de implantação este banco de dados foi obtido como um contêiner para o *Docker*, uma plataforma aberta para desenvolvedores e administradores de sistema criarem, enviarem e executarem aplicações distribuídas (DOCKER, 2018). Por se tratar de um banco de dados relacional, foi construído um diagrama entidade relacionamento, abordando os conceitos de Loja, Produto, Promoção, Oferta e Categoria.

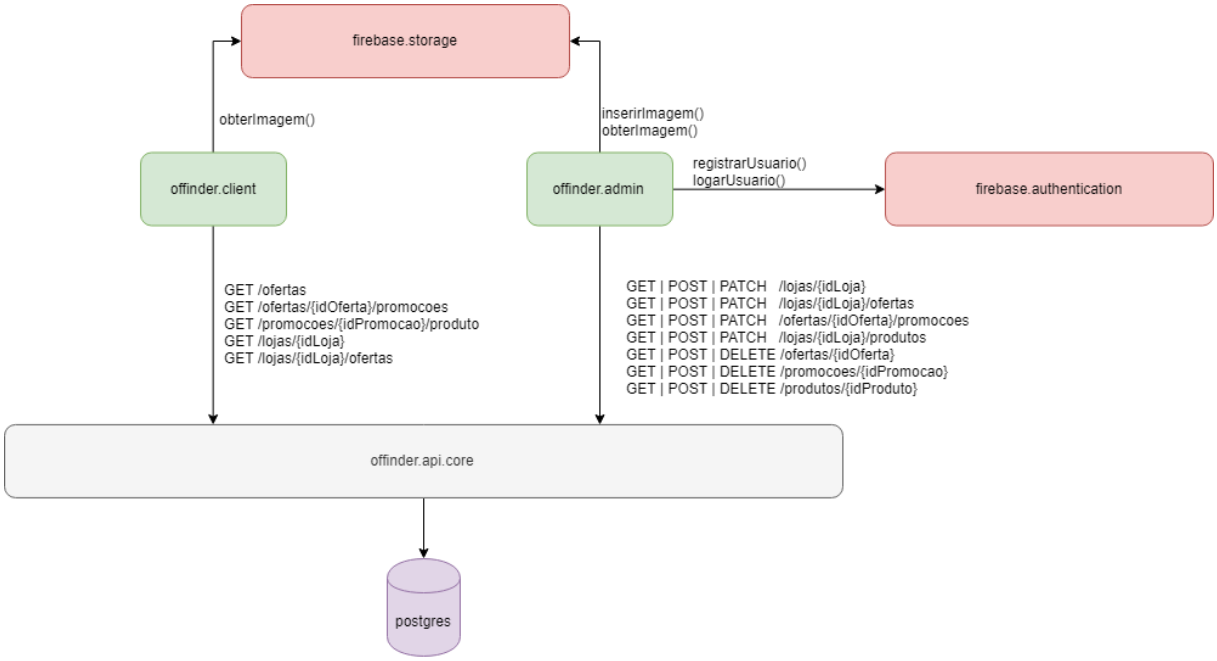
Uma vez definido o banco de dados e sua modelagem, foi iniciado o processo de construção do servidor responsável por fornecer funções sobre ele. Para isto foi definido a utilização de *Web Service*, com o padrão arquitetural REST (*Representational State Transfer*), permitindo uma interoperabilidade entre o cliente e o servidor. Além disso, a abordagem adotada permite que o servidor criado seja utilizado pelos mais diversos tipos de clientes, garantindo que caso seja necessário criar uma aplicação móvel, como foi o caso da plataforma administrativa, esta possa usufruir do serviço fornecido sem nenhuma alteração no servidor.

Para a construção da API REST foi utilizado o *LoopBack*. Este framework foi escolhido após a tentativa de construção utilizando o *Express*, outro framework, porem minimalista para a construção das APIs, assim necessitando de uma maior codificação, consumindo mais tempo neste processo. Além disso, com a utilização do *LoopBack* temos uma suíte de ferramentas de linhas de comando poderosa, que nos permite rapidamente construir cada uma das camadas de nossa aplicação com pouco esforço, e disponibilizá-la via Docker.

Pensando na segurança da aplicação, no armazenamento de imagens, e como forma de facilitar o desenvolvimento de um serviço de recuperação de senha por e-mail, foi adotado o Firebase. Assim, seguindo a arquitetura projetada para esta aplicação, o usuário em uma primeira etapa cria sua conta de usuário através do Firebase Auth, e a partir do *token* gerado no momento do login, garantimos o acesso aos recursos da aplicação. Quando um usuário autenticado deseja cadastrar um novo produto, armazenamos as imagens ilustrativas no Firebase Storage.

Deste modo, a arquitetura geral da aplicação e suas comunicações podem ser vistas na Figura 10:

Figura 10 – Arquitetura Geral da Aplicação e Suas Comunicações



Fonte: o autor

3.4 Codificação

3.4.1 API

Com o projeto definido, iniciou-se o processo de codificação do servidor de aplicação com o Loopback. A implementação desta API segue uma arquitetura baseada em modelos. A primeira etapa do trabalho consistiu na implementação das entidades que modelam os dados da aplicação. Utilizando a ferramenta de linha de comando foram geradas as seguintes entidades:

- Categorias;
- Produtos;
- Promoções;
- Ofertas;
- Lojas.

Com as entidades definidas, utilizando a ferramenta de linha de comando do Loopback foram definidas as relações entre as mesmas, seguindo o diagrama entidade relacionamento definido no inicio do projeto. Com os modelos e relações definidos e apos

configuramos a conexão entre a API e o banco de dados, utilizamos um *script* de migração de dados para construir as tabelas necessárias no banco de dados SQL. Em seguida, foram definidos os repositórios, classes que mapeiam as entidades para o banco de dados conectado a aplicação e por fim as *controllers*, que possuem como responsabilidade expor os métodos presentes nos repositórios, através de *end-points* seguindo o padrão REST . Assim como a camada de modelos e suas relações, as camadas de repositórios e *controllers* foram inicialmente geradas com a ferramenta de linha de comando.

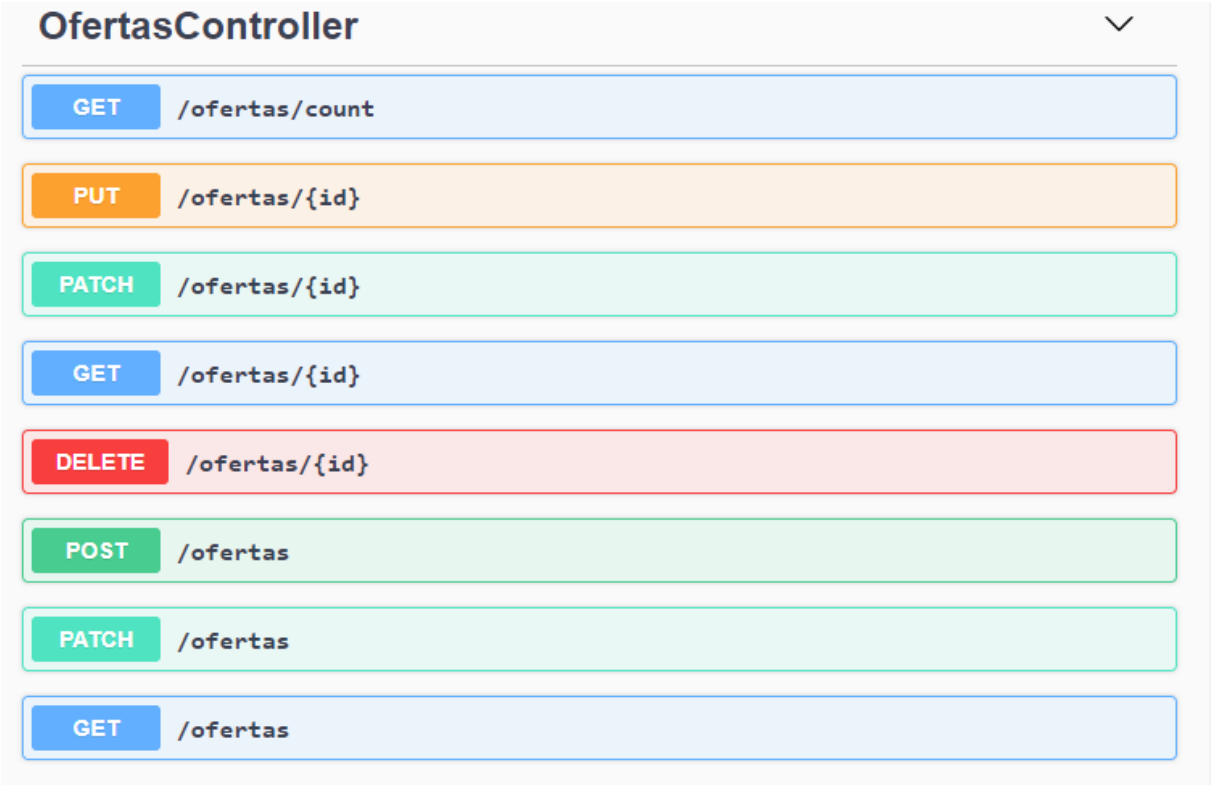
3.4.1.1 *Endpoints*

Nesta seção serão apresentados os principais *endpoints* da aplicação, por se tratar de uma aplicação *RESTful*, para uma mesma rota é possível realizar operações diferentes ao alternarmos os verbos *GET*, *POST*, *PUT*, *PATCH* e *DELETE*.

3.4.1.1.1 OfertasController

Os *endpoints* dessa *controller* estão diretamente relacionados a oferta, permitindo adicionar, editar, remover uma oferta do banco de dados, assim como listar todas as ofertas disponíveis no sistema. Posteriormente, as opções de adicionar, remover e editar ofertas foram removidas, pois a utilização desta parte da API serviria apenas para clientes que desejassem consultar as ofertas no sistema. Os métodos desta *controller* podem ser observados na Figura 11.

Figura 11 – Documentação da API para controladora entidade oferta

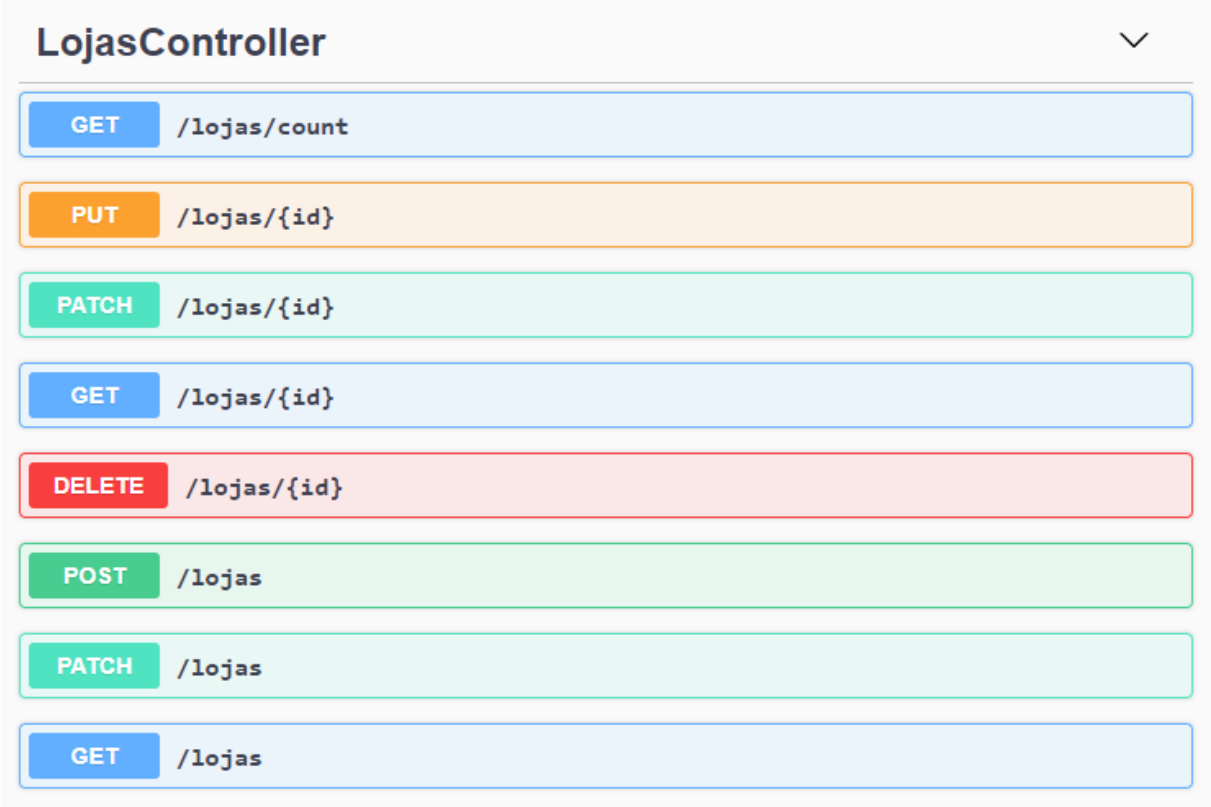


Fonte: o autor

3.4.1.1.2 LojasController

Os *endpoints* dessa *controller* estão diretamente relacionados a entidade loja, permitindo adicionar, editar, remover uma loja do banco de dados, assim como listar todas as lojas existentes no sistema. Os métodos desta *controller* podem ser observados na Figura 12.

Figura 12 – Documentação da API para controladora entidade loja

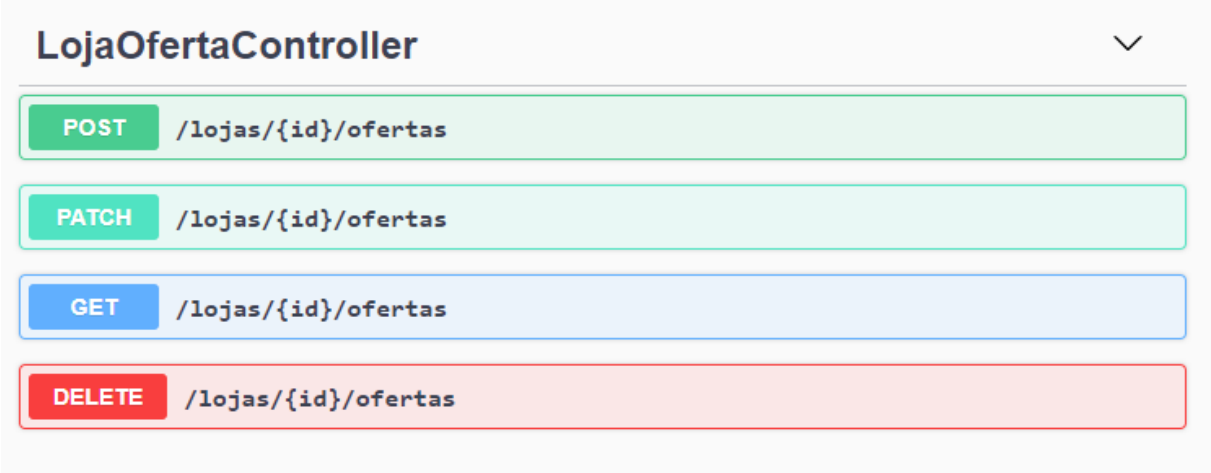


Fonte: o autor

3.4.1.1.3 LojaOfertaController

Os *endpoints* dessa *controller* estão diretamente relacionados a entidade Loja e seu relacionamento um-pra-muitos com a entidade Oferta, permitindo adicionar, editar, remover uma oferta associada Loja do banco de dados, assim como listar todas as ofertas pertencentes a uma Loja. Os métodos *controller* podem ser observados na Figura 13.

Figura 13 – Documentação da API para controladora da relação loja-oferta

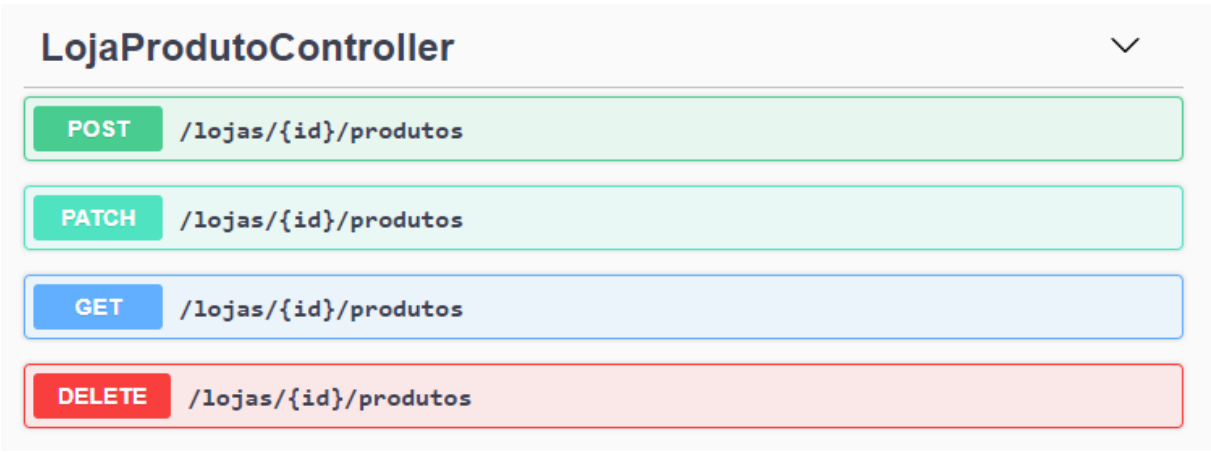


Fonte: o autor

3.4.1.1.4 LojaProdutoController

Os *endpoints* dessa *controller* estão diretamente relacionados a entidade Loja e seu relacionamento um-pra-muitos com a entidade Produtos, permitindo adicionar, editar, remover um produto associada Loja do banco de dados, assim como listar todos os produtos pertencentes a uma Loja. Os métodos *controller* podem ser observados na Figura 14.

Figura 14 – Documentação da API para controladora da relação loja-produto

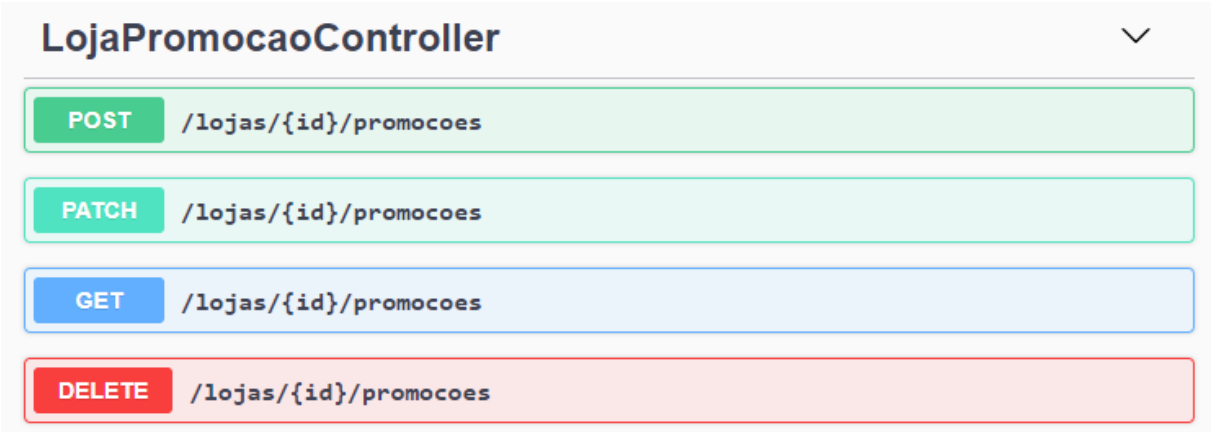


Fonte: o autor

3.4.1.1.5 LojaPromocaoController

Os *endpoints* dessa *controller* estão diretamente relacionados a entidade Loja e seu relacionamento um-pra-muitos com a entidade Promoção. Desta forma, permite-se adicionar, editar, remover uma promoção associada a uma Loja do banco de dados, assim como listar todas as promoções pertencentes a uma Loja. Os métodos *controller* podem ser observados na Figura 14.

Figura 15 – Documentação da API para controladora da relação loja-promoção

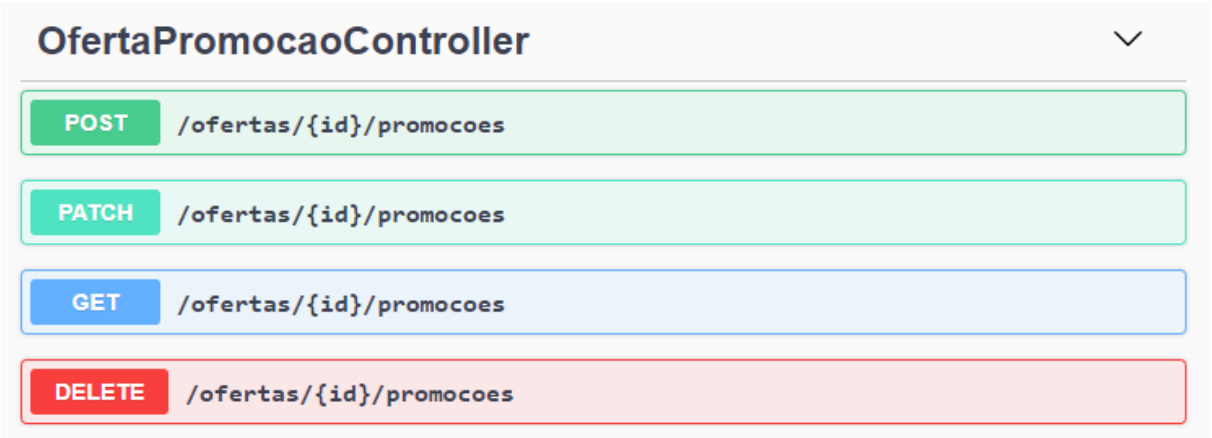


Fonte: o autor

3.4.1.1.6 OfertaPromocaoController

Os *endpoints* dessa *controller* estão diretamente relacionados a entidade Oferta e seu relacionamento um-pra-muitos com a entidade Promoção. Assim, possibilitando adicionar, editar, remover uma promoção associada a uma Oferta do banco de dados, assim como listar todos as promoções pertencentes a uma Oferta. Os métodos *controller* podem ser observados na Figura 16.

Figura 16 – Documentação da API para controladora da relação oferta-promoção

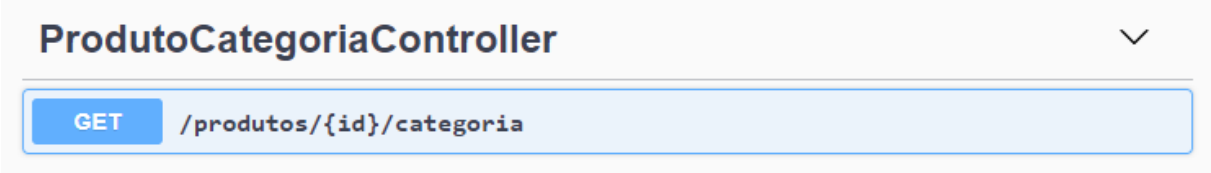


Fonte: o autor

3.4.1.1.7 ProdutoCategoriaController

O *endpoint* dessa *controller* está diretamente relacionado a entidade Promoção e seu relacionamento pertence-a com a entidade Categoria, e permite obter dados referentes a categoria a qual aquele produto faz parte. Os métodos *controller* podem ser observados na Figura 17.

Figura 17 – Documentação da API para controladora da relação produto-categoria

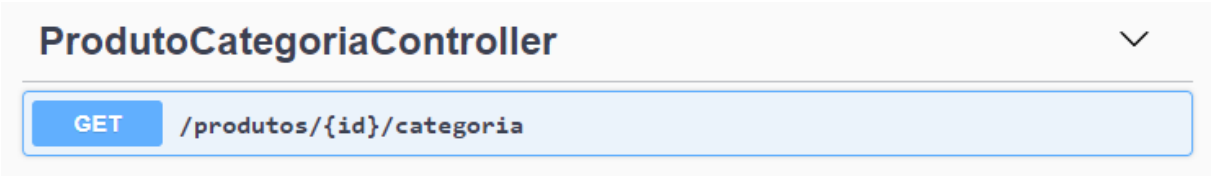


Fonte: o autor

3.4.1.1.8 PromocaoProdutoController

O *endpoint* dessa *controller* está diretamente relacionado a entidade Promoção e seu relacionamento pertence-a com a entidade Produto, e permite obter dados referentes ao produto a qual aquela promoção faz parte. Os métodos *controller* podem ser observados na Figura 18.

Figura 18 – Documentação da API para controladora da relação promoção-produto



Fonte: o autor

3.4.2 Cliente Web

O desenvolvimento do cliente web da solução, partiu do princípio de que a partir dele, seria possível realizar todas as operações do sistema. Então num primeiro momento, o mesmo cliente web poderia ser utilizado em uma consulta realizada por alguém que gostaria de buscar uma oferta no sistema, assim como por algum administrador de uma loja que gostaria de publicar uma oferta no site. Porém, como forma de descentralizar a utilização da plataforma, tentar garantir mais segurança sobre os dados, e testar novas tecnologias, a seção administrativa do site foi descontinuada, e o foco se tornou simplesmente a consulta de ofertas por parte de um cliente.

Para o cliente web, foi utilizado a biblioteca React, e a comunicação com o *back-end*, a partir de chamadas REST com auxilio da biblioteca *axios*.

Os filtros necessários para as consultas são passados como um objeto através dos parâmetros de consulta da requisição.

3.4.3 Aplicação para Administradores

A implementação dessa aplicação surgiu da necessidade de garantir a autenticação do usuário administrativo, e da necessidade de salvar imagens em uma base de dados de armazenamento de arquivos. Para isso a solução encontrada foi utilizar o Firebase.

A escolha do ambiente móvel para o gerenciamento de usuários foi feita devido alguns fatores, como por exemplo, possuir conhecimento sobre o Firebase em aplicações moveis com Flutter. E também para demonstrar que uma mesma API pode ser consumida por múltiplos clientes em ambientes distintos (Web e Móvel) e por como desafio pessoal.

A para a construção dessa aplicação, foi escolhido o padrão Bloc para gerenciar os estados na mesma.

O Bloc é baseado na observação de uma *Stream* de eventos. Por exemplo, ao clicarmos no botão "Minhas Ofertas"adicionamos o evento "Listar ofertas do usuário"na Stream de eventos. Então a camada de negócios da aplicação, observa este evento e tenta mapear um estado. Para este caso em especifico, a aplicação parte do estado inicial "Sem Ofertas", para o estado "Carregando ofertas"enquanto a requisição é processada. Se o resultado do processamento da requisição for uma lista de itens, mudamos o estado para "Ofertas Carregadas", se for uma lista vazia, mudamos o para o estado "Sem ofertas", e por fim caso ocorra um erro no processo, mudamos para o estado "Erro". Durante todo esse processo, a camada de apresentação mapeia o estado atual para algum item na tela, como por exemplo exibir uma animação de carregamento durante o estado "Carregando ofertas".

4 Resultados

Dado a metodologia aplicada neste trabalho, os resultados e a validação da aplicação foram feitos seguindo os casos de teste vinculados a cada uma das historias trabalhadas.

Como exemplo, para o fluxo de autenticação do usuário, foram definidos os seguintes casos de teste:

- Sucesso - Acessar a aplicação com credenciais (e-mail e senha) validos, e previamente cadastrados.
- Erro - Acessar a aplicação com e-mail inválido.
- Erro - Acessar a aplicação com senha inválida.
- Erro - Acessar a aplicação por usuário não cadastrado.

Para o cenário de sucesso a validação a validação ocorreu conforme a Figura 19.

Para o cenário erro por e-mail invalido a validação a validação ocorreu conforme a figura 20.

Para o cenário erro por senha invalida a validação a validação ocorreu conforme a figura 21.

Para o cenário erro por usuário não cadastrado a validação a validação ocorreu conforme a Figura 22.

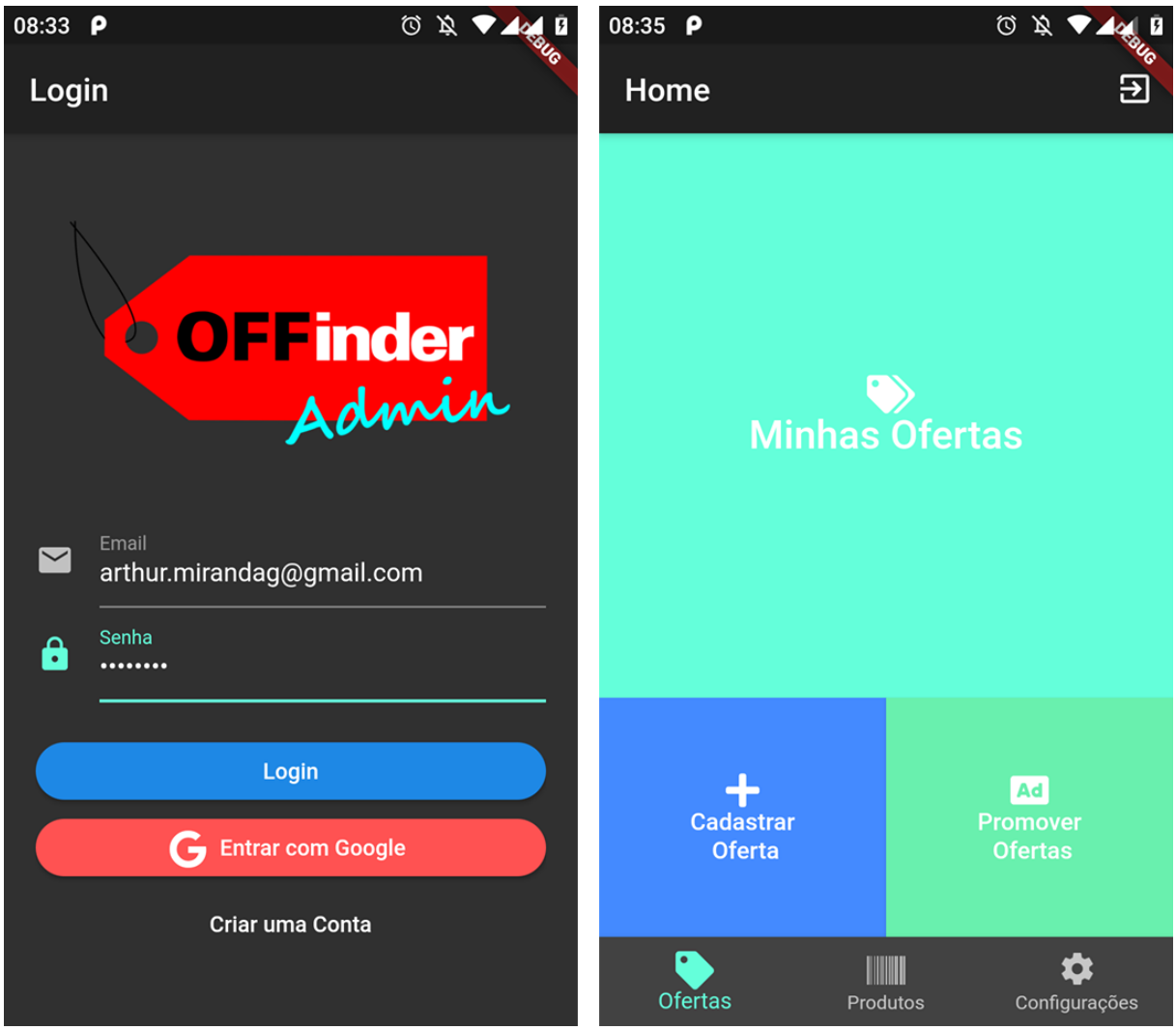
O fluxo de login para novos usuários foi dividido em duas etapas, a primeira efetua o cadastro do usuário no Firebase por meio de e-mail e senha, e a segunda realiza o cadastro do usuário no servidor de *back-end*. Este fluxo é ilustrado pela Figura 23.

O fluxo de cadastro de novos produtos e de novas ofertas ocorrem de forma semelhante. Nestes fluxos o usuário escolhe uma imagem ilustrativa de sua galeria ou pode tirar uma foto com a câmera do celular, e em seguida preenche os dados auxiliares, como nome do produto ou oferta. Estes fluxos são ilustrados pela Figura 24.

O vinculo entre uma oferta e um produto é feito posteriormente na listagem de produtos cadastrados 25, selecionando a oferta para associá-lo e um preço promocional.

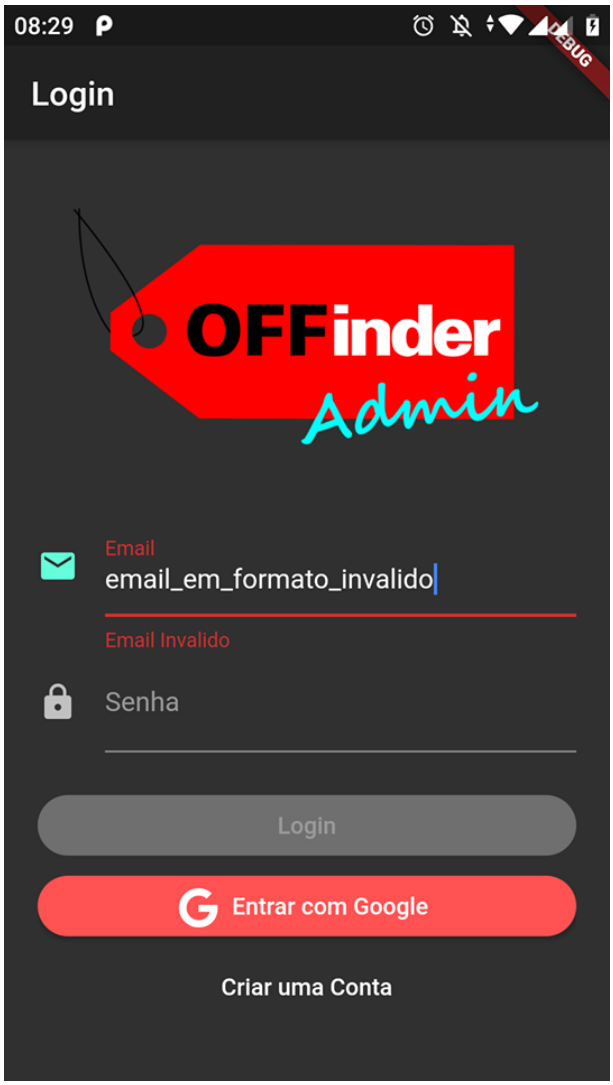
A aplicação web para clientes foi validada da mesma forma, baseando-se nos cenários de teste descritos. Para exemplificar, as telas presentes nas Figuras 26 e 27 foram geradas no fluxo da funcionalidade visualizar ofertas cadastradas.

Figura 19 – Sucesso - Acessar a aplicação com credenciais (e-mail e senha) validos, e previamente cadastrados.



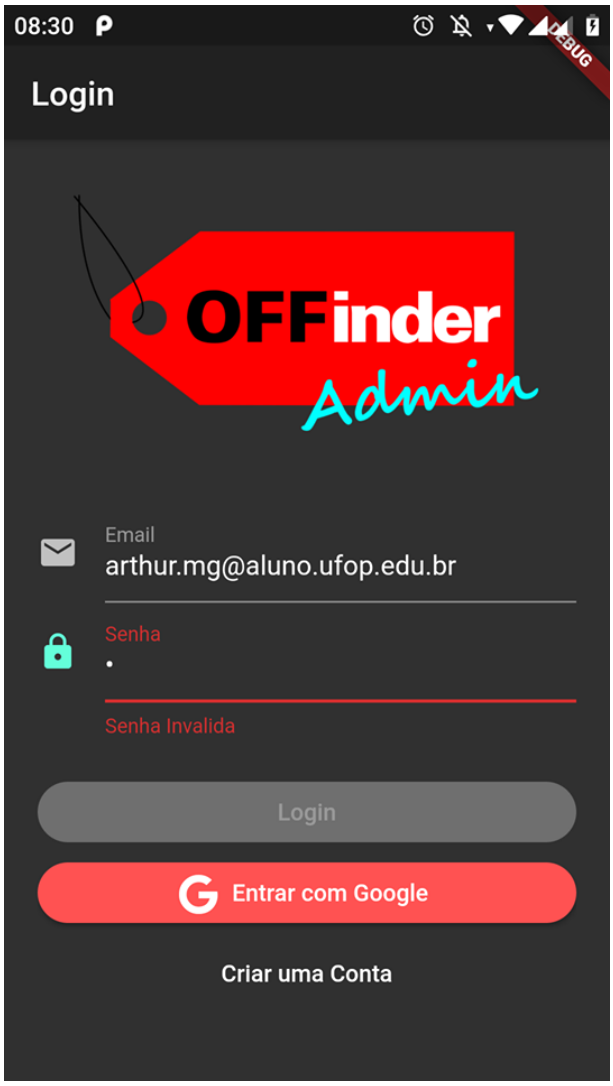
Fonte: o autor

Figura 20 – Erro - Acessar a aplicação com e-mail inválido.



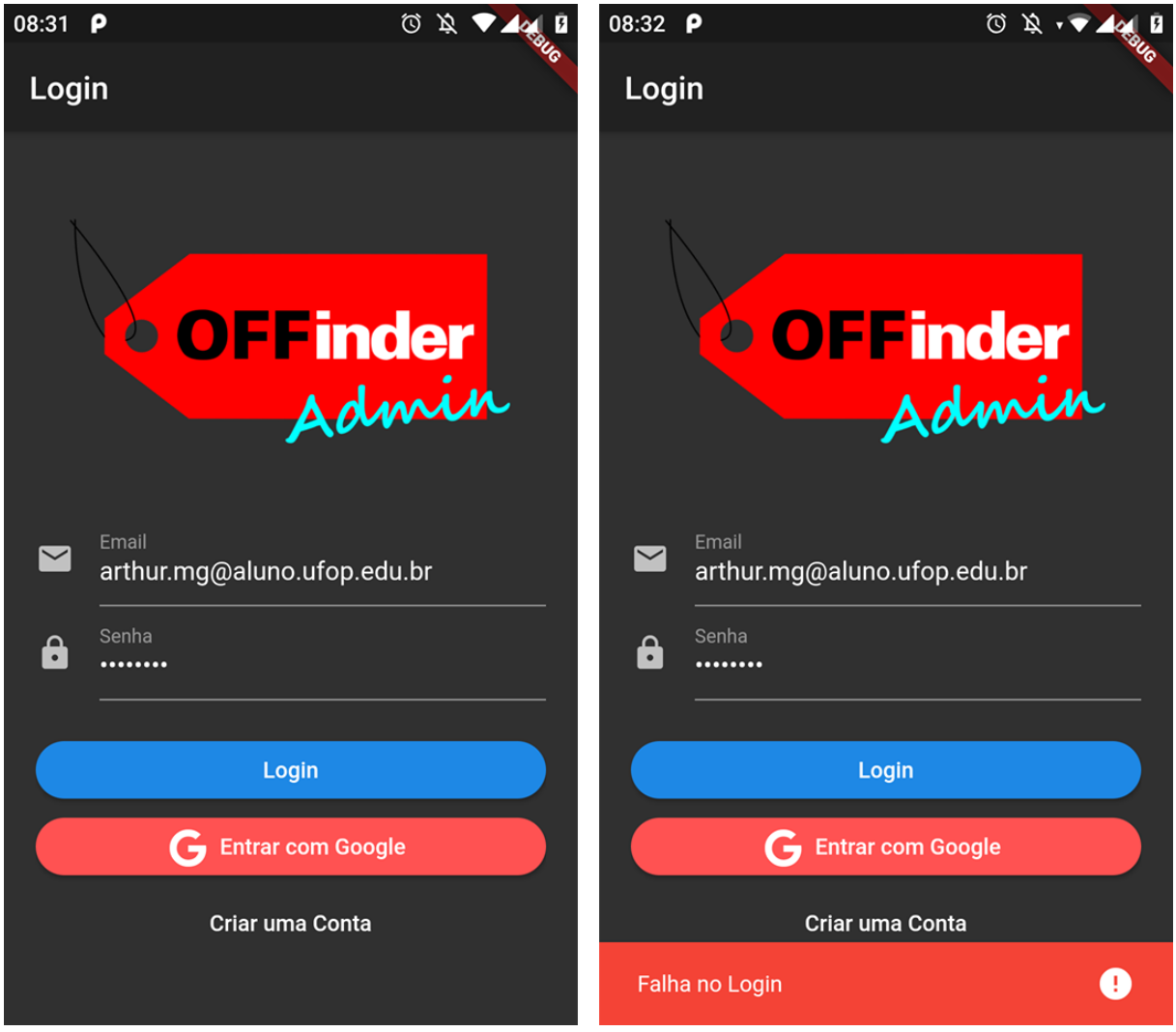
Fonte: o autor

Figura 21 – Erro - Acessar a aplicação com senha inválida.



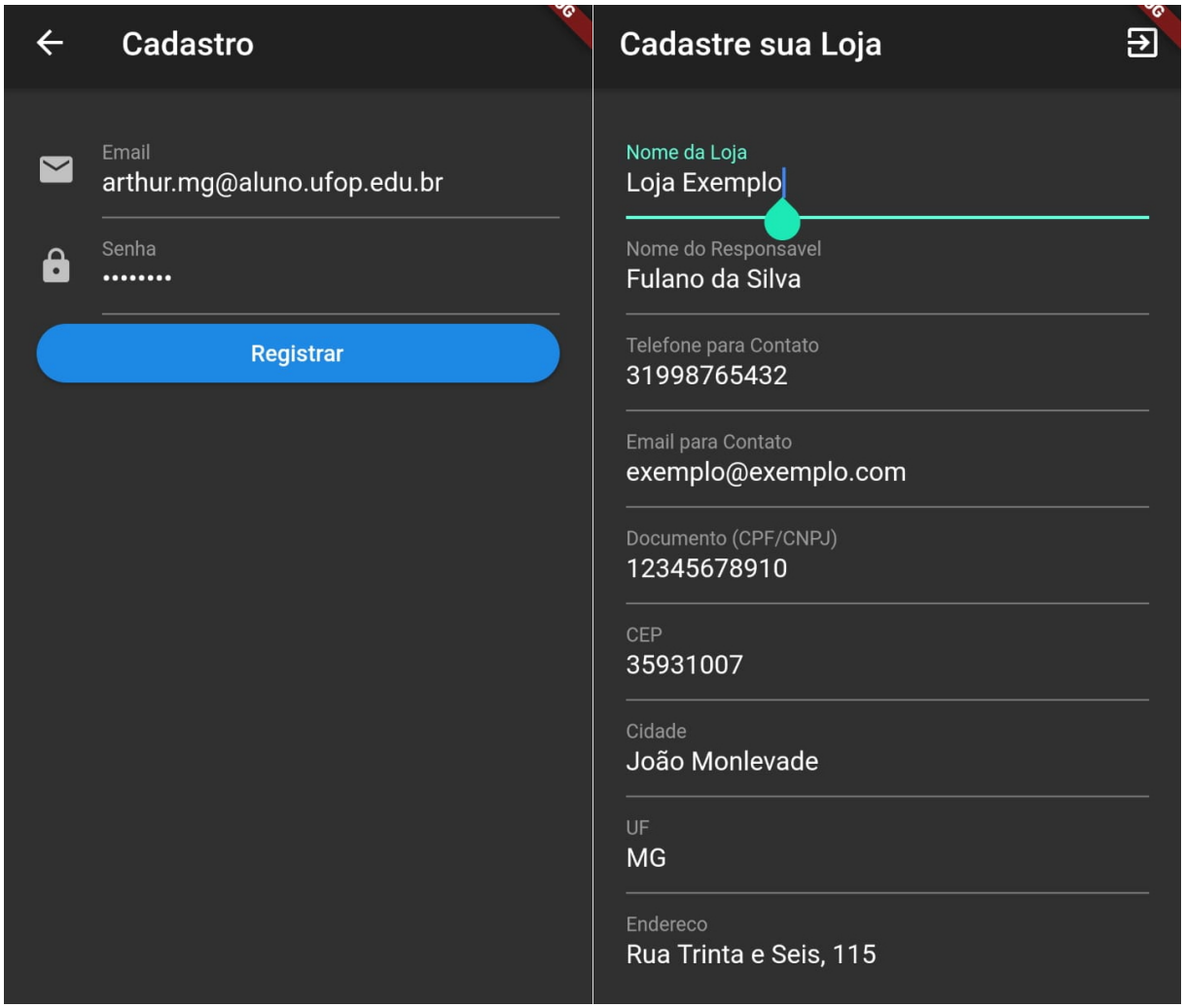
Fonte: o autor

Figura 22 – Erro - Acessar a aplicação por usuário não cadastrado.



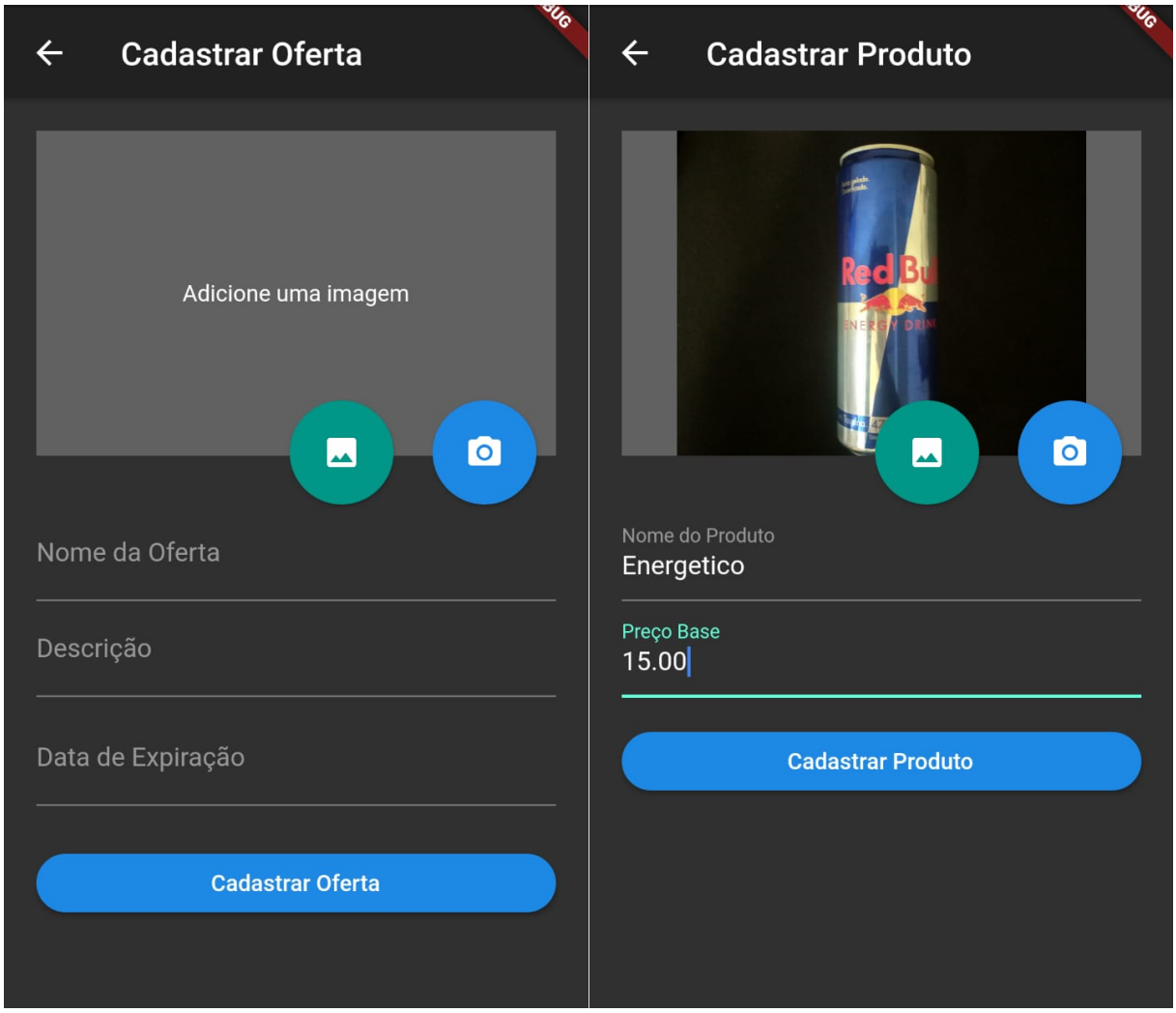
Fonte: o autor

Figura 23 – Tela de registro de novo usuário



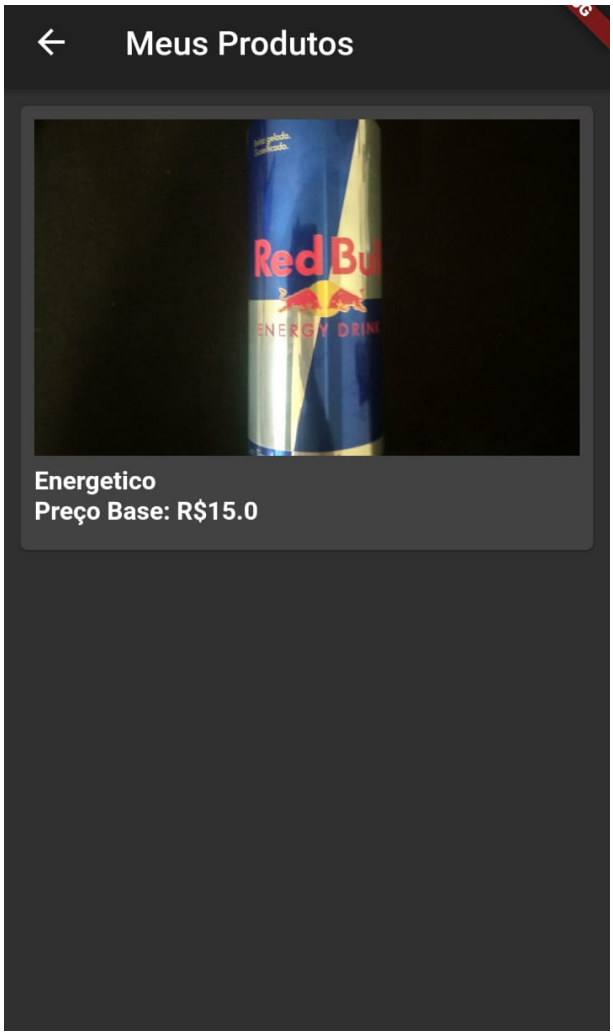
Fonte: o autor

Figura 24 – Tela de cadastro de ofertas e de produtos



Fonte: o autor

Figura 25 – Tela de listagem de produtos cadastrados

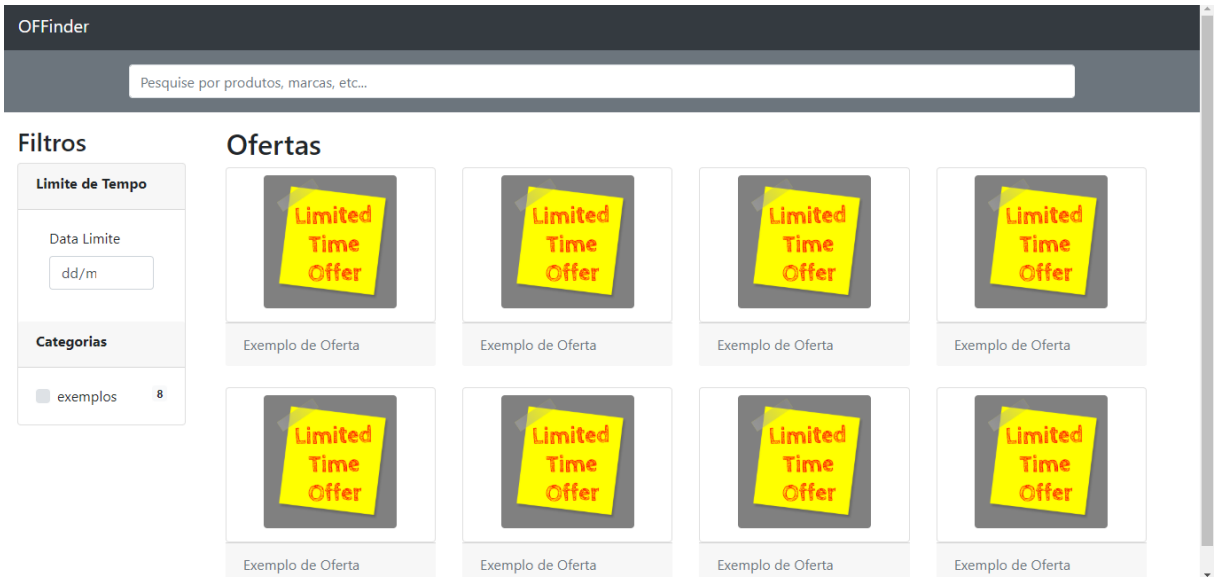


Fonte: o autor

Figura 26 – Tela principal da plataforma web

Fonte: o autor

Figura 27 – Tela de exibição de ofertas válidas



Fonte: o autor

5 Conclusão

Este trabalho apresentou o desenvolvimento de uma plataforma online para divulgação de ofertas, bem como as tecnologias e padrões arquiteturais utilizadas em sua criação.

A plataforma ainda necessita de uma reavaliação quanto a segurança do serviço de *back-end* para então estar pronta para ser testada por empresas reais, uma vez que a segurança dos dados do usuário é um requisito crítico.

Entretanto, podemos concluir que o principal objetivo deste trabalho foi concluído, uma vez que as principais funcionalidades descritas no *backlog* inicial foram implementadas e validadas a partir dos casos de teste descritos inicialmente.

5.1 Considerações Finais

A utilização do Firebase como ferramenta de controle de acesso para o usuário administrativo permite que apenas o usuário registrado tenha acesso a aplicação. Porém para a segurança correta dos dados da aplicação, necessitamos da implementação de algo como um *API Gateway*, de forma que as requisições REST sejam tratadas antes de chegar a API, garantindo assim autenticação e autorização do usuário.

Na aplicação administrativa, na tela de ofertas, foi reservado um espaço para a funcionalidade de "promoção de ofertas", para que um usuário possa solicitar um maior destaque da promoção no site. Porem como ainda é necessária a elaboração de um modelo de negócios para a plataforma, este botão encontra-se desabilitado.

5.2 Trabalhos Futuros

Durante o desenvolvimento do projeto, foram identificadas oportunidades de continuidade e melhorias, considerando limitações de tempo, de ferramentas e conhecimento prático. Assim, podemos citar como trabalhos futuros:

- Melhoria nos aspectos referentes ao *design* e experiência do usuário da aplicação *web*.
- Implementação de uma camada se segurança ao serviço de *back-end* para controle de acesso, utilizando ferramentas de construção de *API Gateway*.
- Análise de negócio para inserção do sistema no mercado.

- Construção de uma aplicação móvel para o ambiente de clientes, de forma a permitir funcionalidades baseadas na geolocalização do usuário.

Referências

ABRANTES, L. *Marketing Digital vs. Marketing Tradicional: qual é melhor?*. 2016. Disponível em: <http://saiadolugar.com.br/marketing-digital-vs-marketing-tradicional/>. Acesso em: 20 junho 2018. Citado na página 15.

ANGELOV, F. *BloC: a predictable state management library for Dart.* 2019. Disponível em: <https://felangel.github.io/bloc/#/>. Acesso em: 05 julho 2019. Citado na página 25.

AONDECONVEM. *Quem Somos*. 2018. Disponível em: <https://corporate.aondeconvem.com.br/quem-somos/>. Acesso em: 20 junho 2018. Citado na página 20.

BECK, K. et al. Manifesto for agile software development. 2001. Citado na página 17.

BOWES, J. *Kanban vs Scrum vs XP – an Agile comparison*. 2015. Disponível em: <https://manifesto.co.uk/kanban-vs-scrum-vs-xp-an-agile-comparison/>. Acesso em: 20 junho 2019. Citado na página 17.

BRASIL, U. *React: o que é e como funciona essa ferramenta?* 2018. Blog da Udacity. Disponível em: <https://br.udacity.com/blog/post/react-o-que-e-como-funciona>. Acesso em: 28 nov 2018. Citado na página 24.

DOCKER. *What is Docker?* 2018. Disponível em: <https://www.docker.com/what-docker>. Acesso em: 20 junho 2018. Citado 2 vezes nas páginas 25 e 28.

FIGMA. *Figma*. 2018. Disponível em: <https://www.figma.com/>. Acesso em: 20 junho 2018. Citado na página 23.

GIL, B. *Quando e porque desenvolver uma SPA (Single Page Application)*. 2016. Vizir. Disponível em: <https://br.udacity.com/blog/post/react-o-que-e-como-funciona>. Acesso em: 28 nov 2018. Citado na página 19.

GUIATO. *2018 Mídia Kit Conectando o seu material a consumidores qualificados*. 2018. Disponível em: <http://corporate.guiato.com.br/media-kit>. Acesso em: 20 junho 2018. Citado na página 19.

MORAIS, R. *Marketing tradicional x Marketing Digital – Entenda as diferenças.* 2013. Disponível em: <http://empreendedorx.com.br/marketing/marketing-tradicional-marketing-digital-entenda-as-diferencas>. Acesso em: 20 junho 2018. Citado na página 15.

PAETSCH, F.; EBERLEIN, A.; MAURER, F. Requirements engineering and agile software development. In: IEEE. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* [S.l.], 2003. p. 308–313. Citado na página 17.

PEREIRA, I. M. *Domain Driven Design: Trabalhando com histórias de usuário*. 2014. DevMedia. Acesso em: 28 nov 2018. Citado 2 vezes nas páginas 18 e 21.

RESULTADOS DIGITAIS. *Marketing Digital.* 2016. Disponível em: <https://resultadosdigitais.com.br/marketing-digital/>. Acesso em: 20 junho 2018. Citado na página 15.

TIENDEO. *O que é Tiendeo?* 2018. Disponível em: <https://www.tiendeo.com/info/pt/>. Acesso em: 20 junho 2018. Citado na página 20.

TERMO DE RESPONSABILIDADE

Eu, **Arthur Miranda Gomes** declaro que o texto do trabalho de conclusão de curso intitulado "***OFFinder - Uma plataforma para divulgação de ofertas do comércio***" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 12 de julho de 2019

Arthur Miranda Gomes